LLNL-PRES-405195

Lawrence Livermore National Laboratory

Classification of HTTP Attacks: A Study on the 2007 ECML / PKDD Discovery Challenge

Brian Gallagher & Tina Eliassi-Rad

gallagher23@llnl.gov eliassirad1@llnl.gov

Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344

HTTP attack classification

- ECML / PKDD 2007 Discovery Challenge
  - http://www.lirmm.fr/pkdd2007-challenge/
  - ECML: European Conference on Machine Learning
  - PKDD: Principles and Practice of Knowledge Discovery in Databases
- Task: Filter application attacks in Web traffic
  1. Recognize an attack
  2. Define which class it belongs to
- Challenges
  - Diversity in attack purposes and means
  - Quantity of data involved and technological shifts
- Data: real-world HTTP query logs

ECML/PKDD 2007 Discovery Challenge Data

Training Data Set

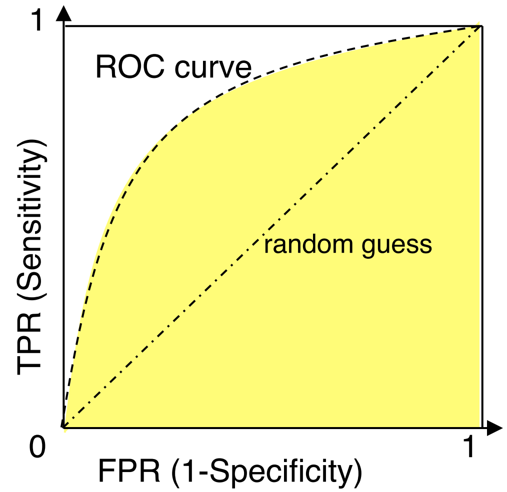
- 50,116 sample HTTP requests
- 15,110 samples (30%) contain attacks
  - Cross-Site Scripting (12%)
  - SQL Injection (17%)
  - LDAP Injection (15%)
  - XPATH Injection (15%)
  - Path traversal (20%)
  - Command execution (23%)
  - SSI attacks (13%)

Test Data Set

- 70,143 sample HTTP requests
- 28,137 samples (40%) contain attacks
  - Cross-Site Scripting (11%)
  - SQL Injection (18%)
  - LDAP Injection (16%)
  - XPATH Injection (16%)
  - Path traversal (18%)
  - Command execution (23%)
  - SSI Attacks (12)

Performance Metrics:
Precision, Recall, F1, Accuracy, and AUC

	Predicted Positive	Predicted Negative
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative



AUC is Area Under the ROC Curve

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Term-frequency based approach

- Treat each HTTP request and attack type as a "bag of terms"
  - Requests are treated as a sequence of terms, separated (i.e. tokenized) by whitespace, '+' characters, and URL encoded characters (e.g., "%20")
- Classify requests based on cosine similarity with attack types

$$P(A = a \mid R) = \alpha \cdot sim_{\cos}(a, R) = \alpha \cdot \frac{\vec{a} \cdot \vec{R}}{\|\vec{a}\| \cdot \|\vec{R}\|} = \alpha \cdot \frac{\sum_{t \in a \cap R} tfidf(t, a) \cdot tfidf(t, R)}{\sqrt{\sum_{t \in a} tfidf(t, a)^2} \cdot \sqrt{\sum_{t \in R} tfidf(t, R)^2}}$$

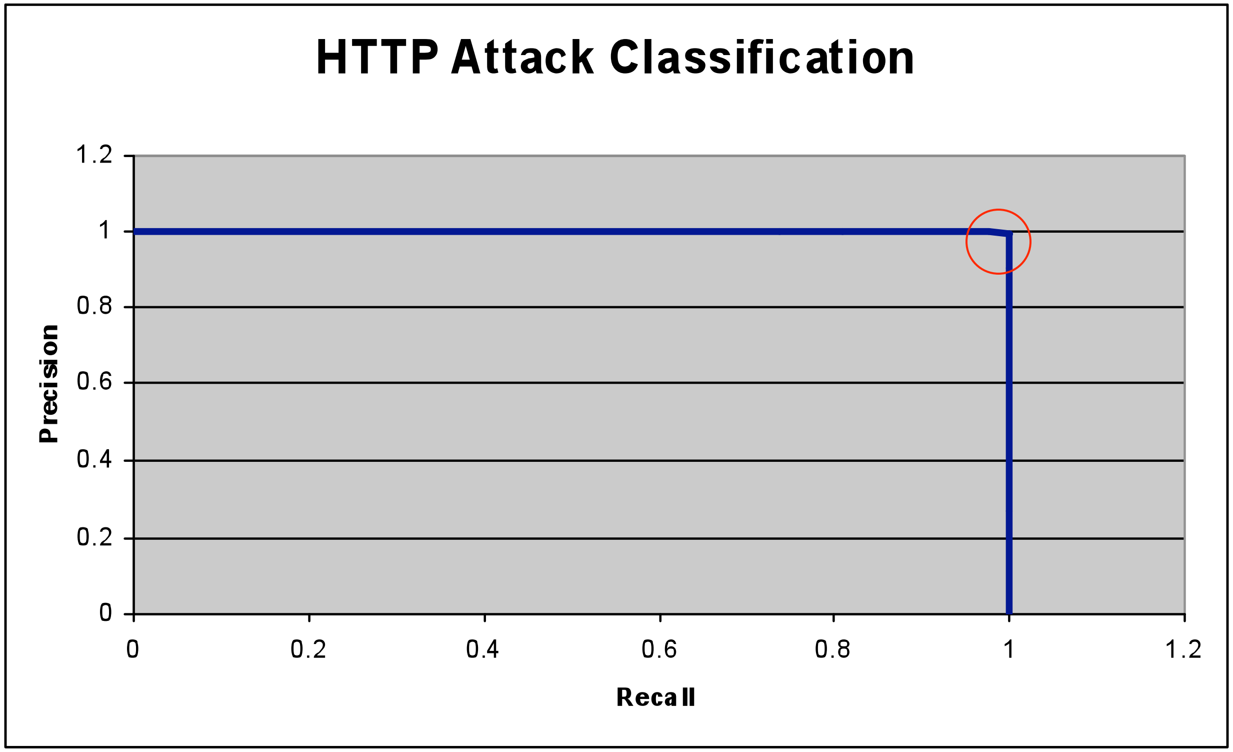
$$tfidf(t, d) = \underbrace{\frac{count(t, d)}{\sum_{v \in d} count(v, d)}}_{\textit{Term Frequency}} \cdot \underbrace{\log \frac{|D|}{\left|\{d_j : t \in d_j\}\right|}}_{\textit{Inverse Document Frequency}}$$

A = random variable for attack types
a = specific attack type
R = incoming HTTP request
count(t,d) = number of occurrences of term *t* in "document" *d*

Salton, Gerard and Buckley, C. (1988). "Term-weighting approaches in automatic text retrieval". *Information Processing & Management* , **24** (5): 513–523.

We achieve over 99% accuracy on training data

- 10-fold cross validation on training set (~50K requests)
- Accuracy = 0.9905, AUC = 0.9990



<u>Decision rule</u>

If $P(A=Valid|R) > \boldsymbol{T}$, then classify R as "Valid"

Otherwise, classify R as:

$$\underset{a}{\arg\max}\, P(A=a)$$

Produce a precision/recall curve by varying $\boldsymbol{T}$

We beat other submissions to the ECML/PKDD Discovery Challenge

- All results reported on labeled training data set
- Competitor 1: Decision Trees (K. Pachopoulos et al.)
  - Accuracy = 0.77
- Competitor 2: Language modeling (M. Exbrayat)
  - Precision = 0.98, Recall = 0.93
  - F1-measure = 0.96
- Our approach: term-frequency based
  - Accuracy > 0.99, AUC > 0.99
  - Precision > 0.99, Recall > 0.99
  - F1-measure > 0.99

ECML/PKDD 2007 Discovery Challenge Data

Training Data Set

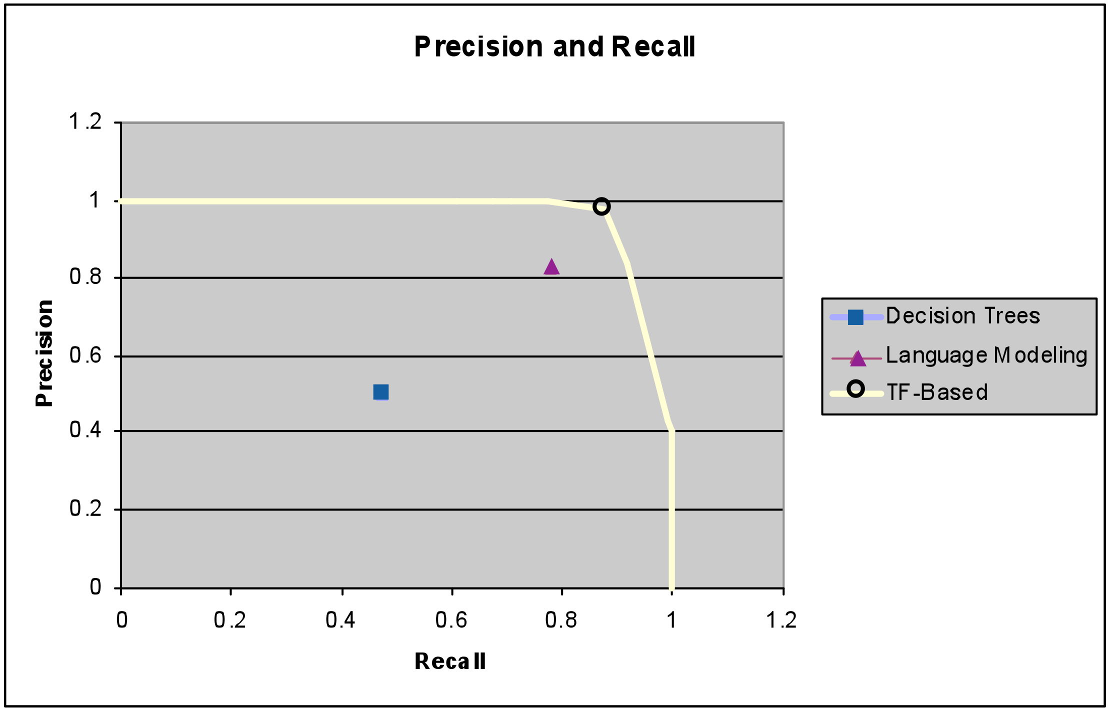
- 50,116 sample HTTP requests
- 15,110 samples (30%) contain attacks
  - Cross-Site Scripting (12%)
  - SQL Injection (17%)
  - LDAP Injection (15%)
  - XPATH Injection (15%)
  - Path traversal (20%)
  - Command execution (23%)
  - SSI attacks (13%)
- 788,559 Unique Terms

Test Data Set

- 70,143 sample HTTP requests
- 28,137 samples (40%) contain attacks
  - Cross-Site Scripting (11%)
  - SQL Injection (18%)
  - LDAP Injection (16%)
  - XPATH Injection (16%)
  - Path traversal (18%)
  - Command execution (23%)
  - SSI Attacks (12%)
- 1,218,553 Unique Terms

Our TF-Based approach is the top performer on the test data too

- Train classifier on training set, test on test set
- Accuracy = 0.94*, AUC = 0.97
- Precision = 0.98, Recall = 0.88, F1 = 0.93

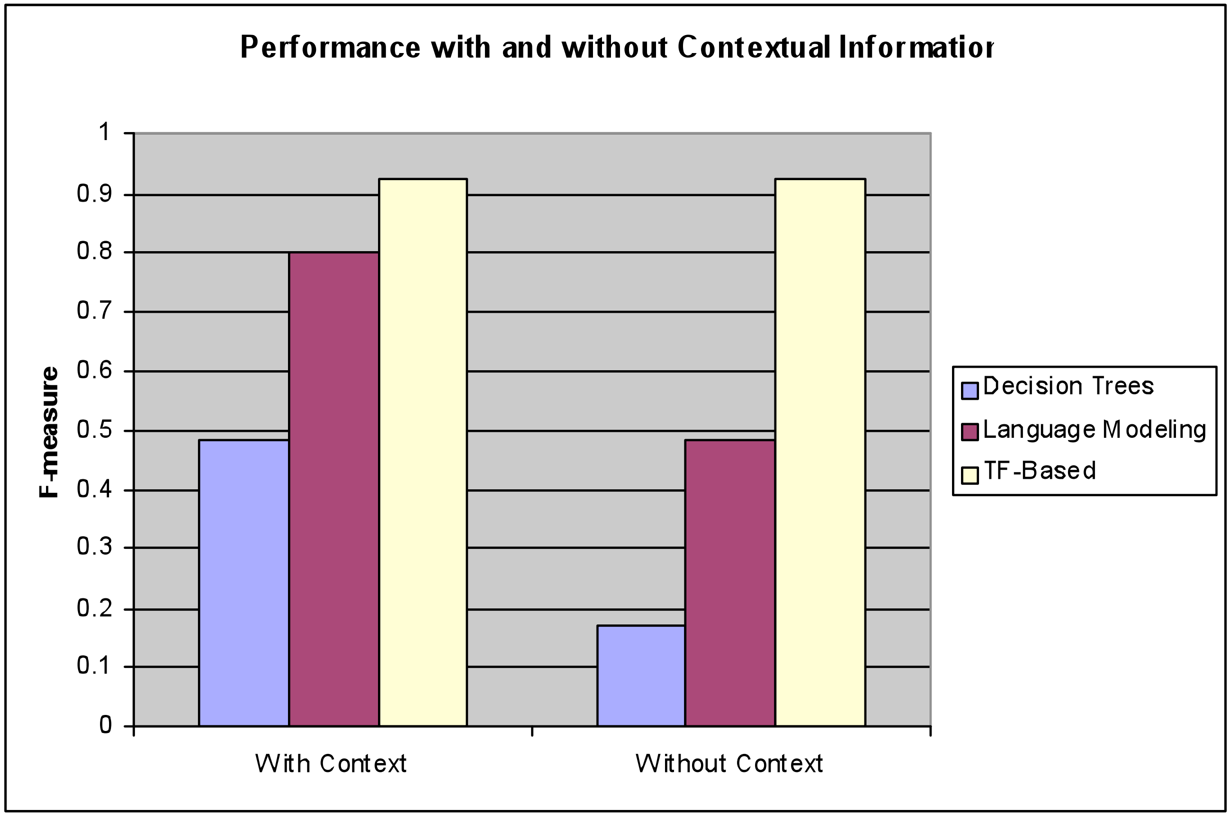


o Decision threshold set based on training data

* 94% of requests are correctly identified as attack vs. non-attack. 91% of requests are correctly classified by type (i.e., "valid" or one of the 7 attack types)

Our TF-Based approach does not rely on attack context

- Other approaches suffer when contextual information is unavailable

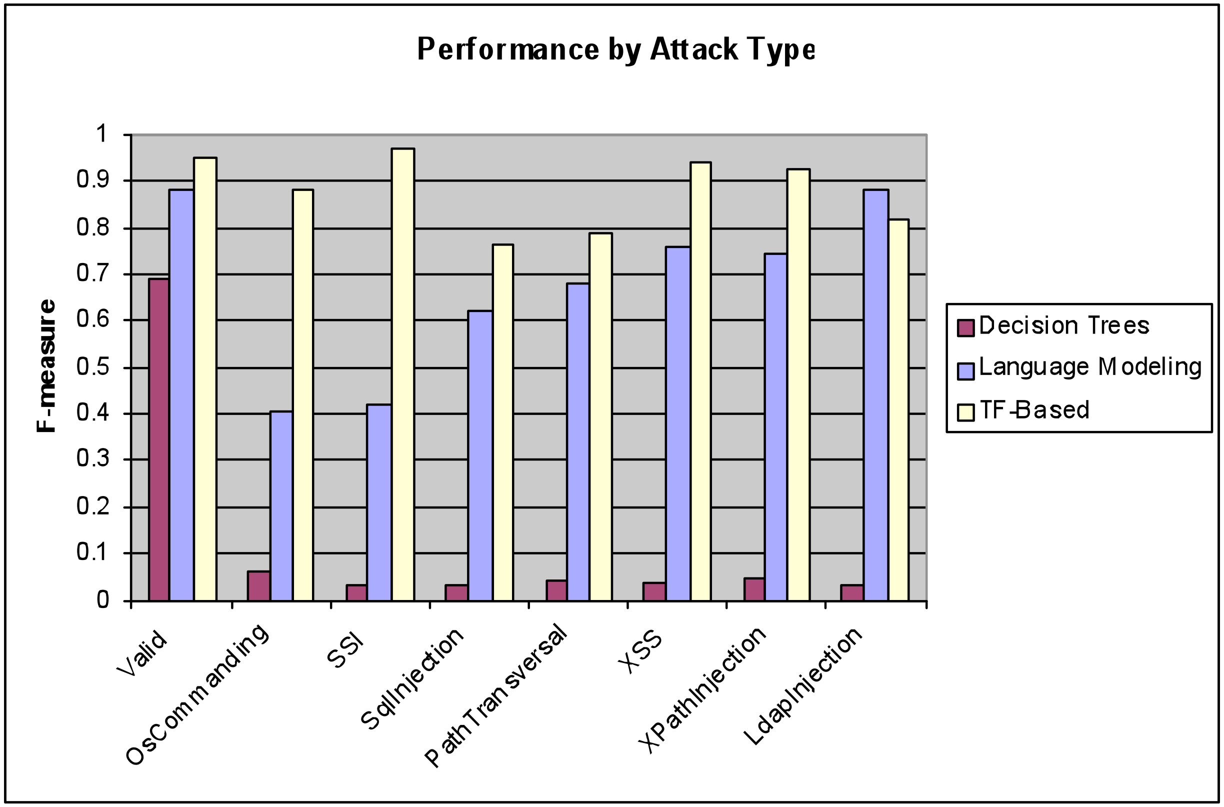


<u>Contextual information</u>

- Operating system running on the Web Server
- HTTP Server targeted by request
- Is XPATH technology understood by the server?
- Is there an LDAP database on the Web Server?
- Is there an SQL database on the Web Server?

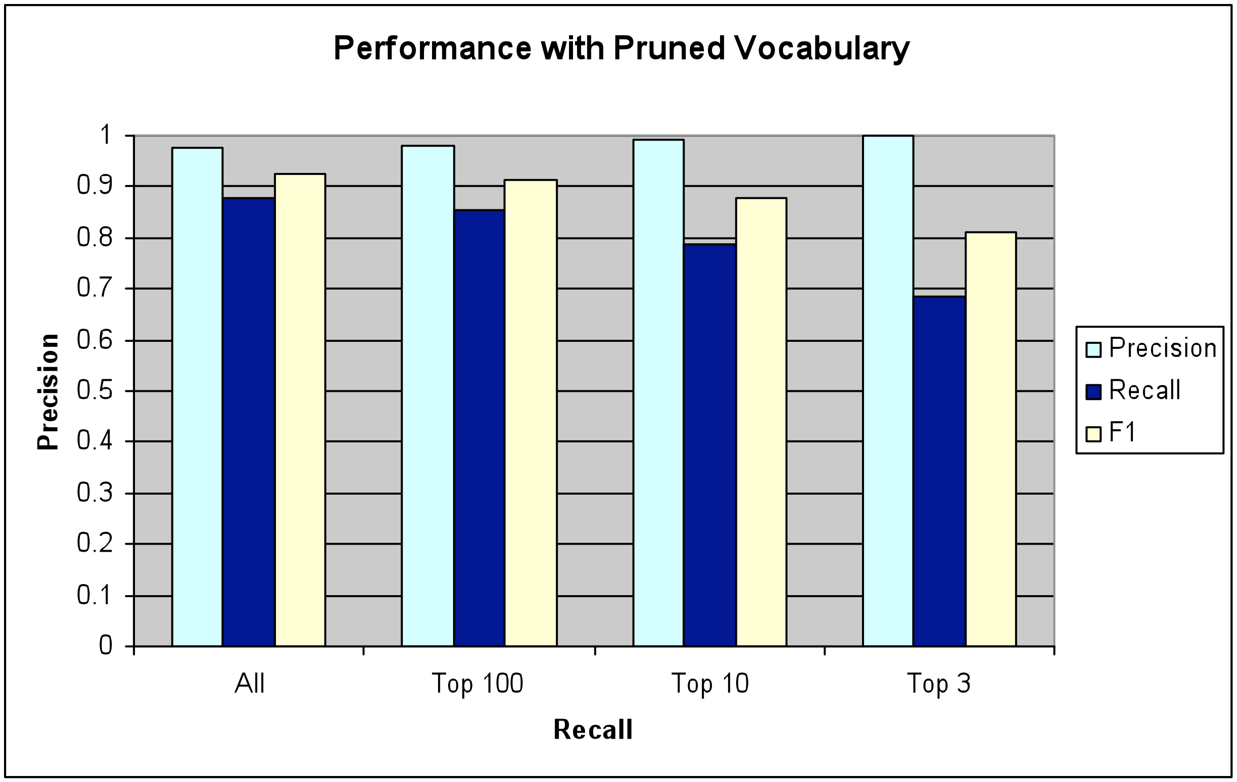
Our TF-Based approach performs consistently across attack types

- Other approaches exhibit greater variance with respect to specific attack types



Our TF-Based approach is robust to aggressive pruning of term vocabulary

- We pruned all but the top-*k* terms for each attack class in the training data, then applied classifier to test data



Our TF-Based approach allows us to characterize attacks by high TF-IDF weight terms – i.e., "keywords"

	Training Data		Test Data	
LDAP Injection	had*	0.005844	had*	0.004065
	objectclass	0.003944	objectclass	0.003861
	*o	0.003872	*o	0.002828
	brien*	0.003872	brien*	0.002828
	netscaperoot	0.001978	displayname	0.002616
Command Execution	..	0.003871	..	0.003558
	dir	0.003546	/c	0.003229
	/c	0.003328	dir	0.002507
	--	0.001650	--	0.001733
	../winnt/system32/cmd.exe	0.001612	../winnt/system32/cmd.exe	0.001678
Path Traversal	..	0.016041	..	0.016415
	.	0.005513	.	0.008600
	virtual	0.002526	virtual	0.001427
	--	0.001713	--	0.000968
	include	0.001263	file	0.000927
SSI Attack	--	0.006241	--	0.006003
	virtual	0.003719	statement	0.002167
	include	0.001859	odbc	0.002167
	statement	0.001275	virtual	0.001967
	odbc	0.001275	progra	0.000810

Our TF-Based approach allows us to characterize attacks by high TF-IDF weight terms – i.e., "keywords"

	Training Data		Test Data	
SQL Injection	**	0.003874	**	0.005199
	select	0.000883	statement	0.001163
	statement	0.000832	odbc	0.001163
	odbc	0.000832	--	0.000807
	union	0.000805	union	0.000674
XPATH Injection	path	0.005394	path	0.005449
	count	0.005108	count	0.005072
	child	0.003756	text	0.002616
	text	0.002421	comment	0.002093
	position	0.002200	child	0.001065
Cross-Site Scripting	document.cookie	0.006498	document.cookie	0.006504
	alert	0.004298	alert	0.004287
	javascript	0.003463	javascript	0.003449
	document.location.replace	0.003209	document.location.replace	0.003208
	url	0.001644	http	0.002411
Valid (No Attack)	13224	0.000061	dddddd	0.002054
	213.191.153.150	0.000057	lkl	0.000969
	9055,045,32	0.000055	large--majorite*des__membres	0.000751
	27260320301	0.000054	ministre-de-l-enseignement-superieur	0.000265
	13.228.134.190	0.000054	tehgghgjty	0.000259

Run-time complexity

- Training time-complexity is $O(|D| \times L_d)$
  - $|D|$ is the number of HTTP query logs in the training set, D
  - L_d is the average length of a HTTP query log in D
- Testing time-complexity is $O(|C|\ L_t)$
  - |C| is the number of attack types + 1 (for the valid HTTP query)
  - L_t is the average length of a HTTP query log in the test set
- Our approach is very efficient overall
  - Linear in the size of a request
  - Proportional to the time needed to read in the data

Summary

- Our approach to HTTP attack classification is very fast (proportional to the time needed to read in the data) and accurate (> 99%)
- We outperform other published approaches on the ECML / PKDD 2007 Discovery Challenge Data
- Our approach automatically characterizes attacks by keywords